Stegräknare

Projekt: Pedometer med AVR

Kurs

Programmering av inbyggda system

2017-04-19

Dennis Bunne LinkedIn

Johan Kämpe LinkedIn

Simon Karlsson LinkedIn

Mjukvaruutvecklare Inbyggda System

MÖLK Utbildning AB

Innehållsförteckning

Figurförteckning

1 Inledning

1.1 Syfte

Syftet med projektet är att tillverka en bärbar, batteridriven stegräknare. Stegräknaren ska ha modulära enheter, det vill säga att de går att ta bort eller byta ut. En modulär enhet kan vara en display, blåtandsmodul eller accelerometer.

En display används för att visa antal tagna steg. Stegen ska avläsas med hjälp av en accelerometer, eller alternativt med en vibrationssensor.

1.2 Noteringar och information

Projektet utfördes på under perioden mars till april, 2017.

I rapporten hänvisas själva stegräknaren ofta till som *projektet* eller *produkten*.

I rapporten hänvisas MPU-9250-enheten, som används för att mäta acceleration, ofta som *MPU-enheten*, eller *MPU-n*.

1.3 Bakgrund

1.3.1 Definitioner av begrepp och förkortningar

Term	Förklaring
Acceleration	Förändring av hastighet per tidsenhet. Kan vara positiv (hastighetsökning) eller negativ (minskad hastighet).
Arduino UNO	Utvecklingskort.
Atmega328p	En mikrokontroller (MCU) med 28 anslutningar, från tillverkaren Atmel
AVR	Familj av Mikrokontrollers som tillverkas av företaget Atmel.
Bibliotek	En samling av kod-funktioner och/eller definitioner.
Breadboard	Kopplingsplatta, används för att ansluta enheter till varandra.
C	Det programmeringsspråk som används i projektet.
DIL	Dual in-line. Standard för mikrochip med två rader av ben.
Git	Ett versionshanteringssystem
GitHub	Molnlagring för Git-projekt
I²C	Ett sätt att koppla samman enheter, via två ledare: SCL och SDA.
IDE	*Integrated Development Environment*. Ett datorprogram eller en programsvit som vanligtvis innehåller en texteditor, kompilator och debugger. För att underlätta vid programmering.
Interrupt	En del i en mikroprocessors funktionalitet. Ett sätt att få processorn att avbryta sitt nuvarande arbeta och övergå till en annan uppgift.
ISP	*In-system programming*, programmering av redan installerade eller inbyggda system.
LED	*Light Emitting Diode*, lysdiod.
Lib	Se *Bibliotek.*
Library	Se *Bibliotek.*
Macro	En definierad konstant i programkod.
MCU	Microcontroller Unit, se *Mikrokontroller.*
Mikrokontroller	En liten enchippsdator med CPU, arbetsminne och programminne.
Mikroprocessor	Processor i ett chip
MPU	Microprocessor Unit, Se *mikroprocessor*
MPU-6050	Mikroprocessor med sensor för gyro och acceleration i tre riktningar vardera.
MPU-9250	Mikroprocessor med sensor för gyro, acceleration och kompass i tre riktningar vardera.
Pull up-reistor	Används i logiska kretsar för att se till att en ingång håller sig hög när den inte har någon anslutning.
Reistans	Värdet på ett motstånd / resistor. Anges i Ohm (Ω).
Resistor	Motstånd i elektriska kretsar.
SCL	*Serial Clock Line*, se *I²C.*
SDA	*Serial Data Line,* se *I²C.*
Serial Monitor	Funktion i Arduinos IDE, för att kommunicera med inkopplad Arduino-enhet. Används ofta för debugging.
Standardbibliotek	Kodbibliotek som alltid är tillgängliga för ett specifikt programspråk. Exempelvis *stdio.h* för programspråket C.
Stegräknare	En enhet som mäter hur många steg en människa tar
TWI	*Two Wire Interface*, se *I²C.*
Utvecklingsmiljö	Se *IDE.*

1.3.2 Stegräknare

En stegräknare är en vanligtvis portabel enhet. Som mäter hur många steg en människa tar. Den kan användas vid exempelvis löpning, promenad, eller för att se hur många steg en person tar under en dag.

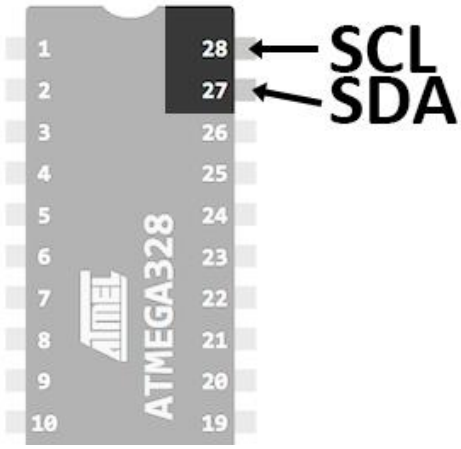
En stegräknare kan också kallas pedometer.

1.3.3 I²C

I²C, *Inter-Integrated Circuit*, är ett sätt att koppla enheter till moderkort, inbyggda system, mobiltelefoner eller andra enheter.

Kommunikation sker via två ledare, SDA (*Serial Data Line*, datasignal) och SCL (*Serial Clock*, klocksignal). Flera I²C-enheter kan anslutas till samma SDA- och SCL-ledare.

På en mikrokontroller av typen Atmega328p, är anslutningar 27 och 28, SDA och SCL, respektive.



Figur 1 I²C-ansluting på Atmega328

I²C kallas även TWI, *Two Wire Interface*, eller *IIC*.

1.3.4 Pull up-resistorer

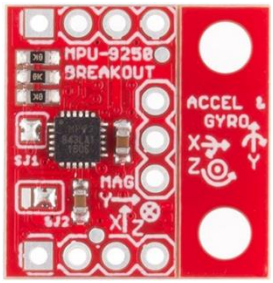
Pull up-resistorer används i logiska kretsar för att se till att en ingång håller sig hög när den inte har någon anslutning.

Utan pull up kan ingången anta vilket läge som helst, hög eller låg. I projektet används pull up-resistorer för I²C-anslutningarna.

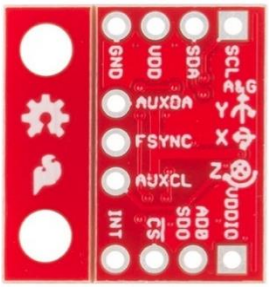
Resistorer som används ska vara av hög resistans.

1.3.5 MPU-9250

Enheten som används i projektet för att mäta acceleration kallas *MPU-9250 IMU Breakout* och är tillverkad av Sparkfun. På enheten finns en MPU-9250.



Figur 2 MPU-9250, framsida



Figur 3 MPU-9250, baksida med anslutningsbeteckningar

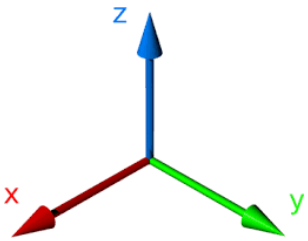
De anslutningar som används i projektet är VDD och GND för drivning, samt SDA och SCL för I²C-kommunikation.

Länk till tillverkarens produktsida finns i kapitel 1.4.4

1.3.6 X-Y-Z-riktning

X, Y och Z används inom projektet för att beskriva riktningar i en tredimensionell miljö. MPU-9250-enheten har sensorer för acceleration i dessa tre riktningar.

När enheten förs åt ett visst håll mäts rörelsens acceleration i denna riktning.



Figur 4 Riktningar i X- Y och Z-led

1.4 Länkar

1.4.1 GitHub

Länkar till projektets GitHub-sida

GitHub-sida för projektet
https://github.com/GoblinDynamiteer/stepCounter

GitHub-sida, huvudkod
https://github.com/GoblinDynamiteer/stepCounter/blob/master/Kod/Huvudkod/stepCounter/mpu_conntest/main.c

GitHub-sida, kod för prototyp 1
https://github.com/GoblinDynamiteer/stepCounter/blob/master/Kod/Testkod/Arduino/stepCounterPrototype/stepCounterPrototype.ino

1.4.2 Använda funktionsbibliotek

Länkar till bibliotek som har använts i projektets kod

MPU 6050 library
http://davidegironi.blogspot.se/2013/02/avr-atmega-mpu6050-gyroscope-and.html#.WOtTeojyhPY

U8glib
https://github.com/olikraus/u8glib

I2C master library
http://homepage.hispeed.ch/peterfleury/avr-software.html#libs

1.4.3 Datablad

Atmega328p
https://github.com/GoblinDynamiteer/stepCounter/blob/master/Dokumentation/Komponenter/Atmega%20328p/ATmega328P_datasheet_Complete.pdf

MPU-9250
https://github.com/GoblinDynamiteer/stepCounter/blob/master/Dokumentation/Komponenter/MPU%209250/MPU-9250-Datasheet.pdf

SSD1306 Display
https://github.com/GoblinDynamiteer/stepCounter/blob/master/Dokumentation/Komponenter/Micro%20OLED%20Breakout%20Display%20SSD1306/SSD1306_datasheet.pdf

1.4.4 Andra länkar

MPU-6050 Accelerometer + Gyro
Kodexempel
http://playground.arduino.cc/Main/MPU-6050

SparkFun IMU Breakout - MPU-9250
Produktbeskrivning
https://www.sparkfun.com/products/13762

2 Genomförande och resultat

2.1 Använd programvara

Programnamn	Beskrivning
Atmel Studio 7.0	IDE för att skriva kod och kompilera till Atmel MCU.
Atom	Texteditor, för att skriva och analysera kod.
Git	Versionshanteringsverktyg.
GitKraken	Grafiskt gränssnitt (GUI) för Git.
Fritzing	Program för att skapa kretsscheman.
Microsoft Word	Dokumentering och rapportskrivning.
Google Docs	Rapportskrivning.
AVRDUDE, AVR Downloader/UploaDEr	Program för att ladda upp kompilerad kod till MCU.
AVRDUDESS	Grafiskt gränssnitt (GUI) för AVRDUDE.
Autodesk Fusion 360	CAD-program för att konstruera ett hölje till stegräknaren.

2.2 Använd hårdvara

Komponent	Beskrivning
Atmel ATMega328-PU DIP-28N 8-bit MCU	Mikrokontroller.
SparkFun MPU-9250	Enhet som mäter acceleration i X, Y och Z-riktning. Innehåller även gyroskop, temperatur och kompass-funktionalitet.
Micro OLED Display SSD1306 128x64	Display med upplösning 128x64 pixlar.
AVR pocket programmer	Enhet för att programmera Atmega-MCU via en dators USB-port.
DIL-hållare 28-pin 0.3"	Sockel för Atmega-MCU.
Resistor 4,7kΩ, 2 st.	Pull up-resistorer för I²C-kanaler.
Kristalloscillator 16 MHz	Kopplas till MCU för att öka klockfrekvensen från 1 MHz till 16 MHz.
Kondensator 22pF, 2 st.	Används för inkoppling av kristalloscillator.
Litiumbatteri 18650	Strömförsörjning.
Hållare för litiumbatteri 18650	
Grovekontakter med 4-stift	För modularisering av I²C-komponenter.
Kopplingsplatta 60x40 mm	För sammanlödning av komponenter.
Arduino UNO	För prototyping.

2.3 Metod

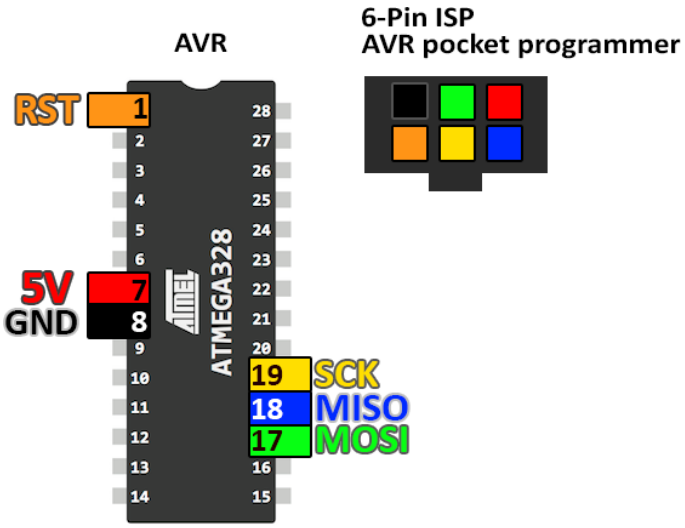
2.3.1 Skrivning av källkod, kompilering och uppladdning till mikrokontroller

Programmets källkod skrevs i utvecklingsmiljön Atmel Studio, som ett *GCC C Executable Project*-projekt. Källkoden är skriven i programspråket C.

All egenskriven kod finns i projektets källkodsfil *main.c*.

Vid kompilering av koden skapas en fil med filändelsen *hex* i Atmel Studio-projektets *Debug*-underkatalog. Denna fil laddas upp till mikrokontrollern med datorprogrammet AVRDUDE och programmeringsenheten *AVR pocket programmer*.

Programmering utförs via ISP, följande anslutningar på mikrokontrollerns används.



Figur 5 Inkoppling av programmeraren till mikrokontrollern

Programmeringsenheten kan även användas som strömkälla.

2.3.2 Studie av datablad

Tidigt i projektet samlades datablad in för införskaffad hårdvara. Databladen studerades och relevant information noterades för att kunna användas vid framtagning av prototyp.

I synnerlighet studerades datablad för:

- **MPU-9250**
- **Atmega328p**
- **OLED Display**

2.3.3 Studie av funktionsbibliotek

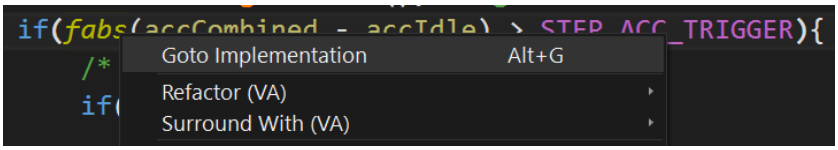
Tidigt i projektet hämtades färdiga kodbibliotek från Internet, för projektets hårdvara. Koden i dessa bibliotek studerades, eller användes för att undersöka funktionen hos olika komponenter.

Exempel på bibliotek som studerades eller användes i projektets utveckling:

Bibliotek	Beskrivning och syfte
Wire.h	I^2C-bibliotek för Arduino-enheter. För att funktionstesta I^2C-enheter med Arduino UNO. Användes även i koden för prototyp 1.
twitest.c	I^2C-test/demo för AVR-enheter.
mpu6050.h	MPU-6050 (Accelerometer, mm.) för AVR-enheter.
i2cmaster.h	I^2C-bibliotek för AVR-enheter.
ssd1306.c	Kod som använt sig av **i2cmaster.h** för att skriva ut text till display.

Koden studerades med bland annat texteditorn Atom och i utvecklingsmiljön Atmel Studio.

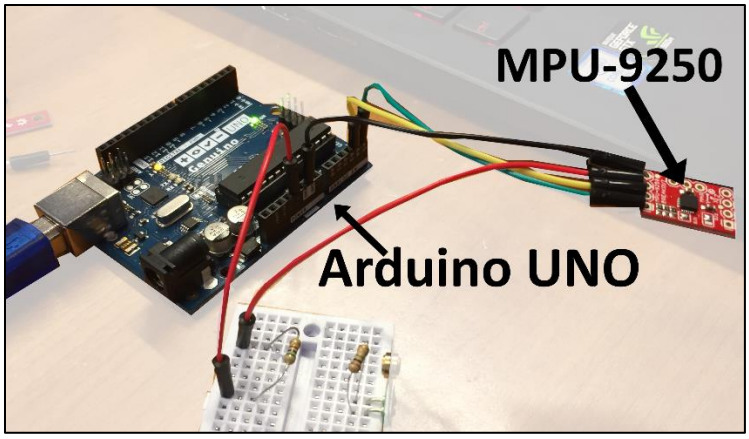
I Atmel Studio finns möjligheten att automatiskt gå till den fil där ett makro, en funktion eller en variabel är definierad. I bilden nedan utförs detta på funktionen *fabs()*;



Figur 6 Gå till en implementation av ett makro, variabel eller funktion.

2.4 Prototyp 1

Projektets första prototyp, kallad *prototyp 1*, byggdes för att säkerställa att en stegräknare kunde konstrueras med en MPU-9250 som accelerometer. MPU-enheten kopplades direkt till en Arduino UNO-enhet via en breadboard.

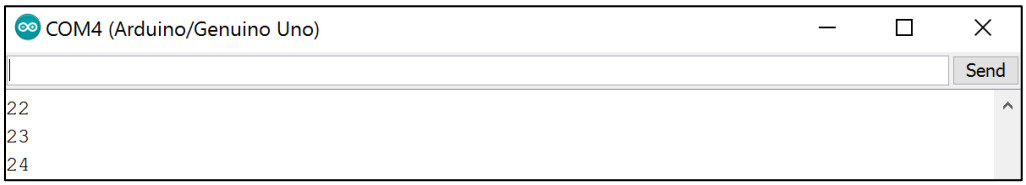


Figur 7 Uppkoppling av MPU-9250 till Arduino UNO

Koden för prototyp 1 baserades på kod skapad av JohnChi [1], som skriver ut värden från MPU-6050-enheter till Arduino-utvecklingsmiljöns *Serial Monitor.* Koden skrevs om så att endast värden för acceleration hämtades, dessa användes för stegräkning.

Funktionsbiblioteket **wire.h** används för att ansluta MPU-n via I^2C till Arduino-enheten. Portarna A4 och A5 används för SDA och SCL, respektive. En resistor med svärdet 560 Ohm används för att ta ner spänningsmatningen till MPU-n från 5V till ca 3V.
Enligt tillverkarens hemsida[2] ska enheten drivas med 2.4V – 3.6V.

En bärbar dator användes för strömförsörjning och för att avläsa *Serial Monitor*. Avläsning av steg gjordes genom att placera prototypen i en byxficka och promenera runt.



Figur 8 Utskrift av stegräkning till Serial Monitor

En vibrationssensor testades också för att avläsa steg. Även om den kunde användas för avläsning, valdes den att inte användas, då MPU-enheten ansågs vara mer pålitlig.

Prototyp 1 färdigställdes 2017-03-21

Kopplingsschema för prototyp 1 kan ses i rapportens bilagor.

[1] Se länk i kapitel 1.4.4
[2] Se länk i kapitel 1.4.4

2.5 Prototyp 2

Vid byggnation av *prototyp 2* användes en Atmega328p MCU.

En breadboard användes för samtliga inkopplingar innan sammanlödning av enheter påbörjades.

Initialt anslöts MPU-9250-enheten till mikrokontrollerns I^2C-anslutningar, med pull up-resistorer på båda kanaler. För att kunna indikera att MPU-n fungerade korrekt användes en blå LED. Lysdioden sattes att lysa när accelerationsdata från MPU-n översteg ett visst värde.

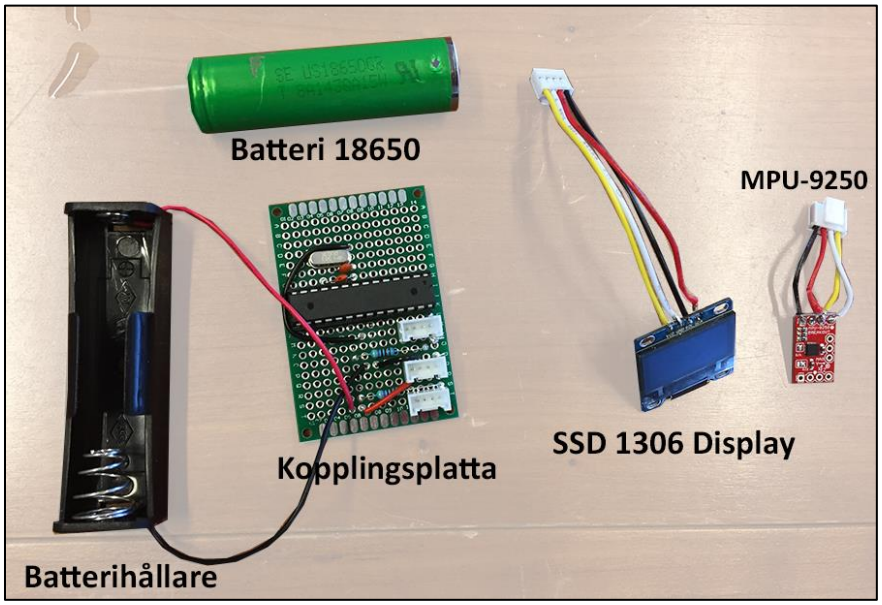
När displayen anslöts och kunde användas för att visa diverse data från MCU-n, blev lysdioden överflödig och användes inte längre.

En kristalloscillator på 16 MHz kopplades MCU-enhetens anslutningar 9 och 10. Dessa anslutningar används för inkoppling av externa kristalloscillatorer. Kristalloscillatorn användes för att öka klockfrekvensen på MCU-n. Enheten upplevdes också som stabilare.

När uppkopplingen på breadboard ansågs vara fungerande löddes MCU-n fast via en sockel på en kopplingsplatta. Tre stycken kontakter löddes till MCU-ns I^2C-kanaler. Dessa kontakter går att använda för att koppla in olika enheter med I^2C-stöd.

På displayen och MPU-enheten löddes kontakter fast, för anslutning till kontakterna på kopplingsplattan.

På kopplingsplattan löddes även övriga komponenter fast: kristalloscillator, resistorer och kondensatorer.



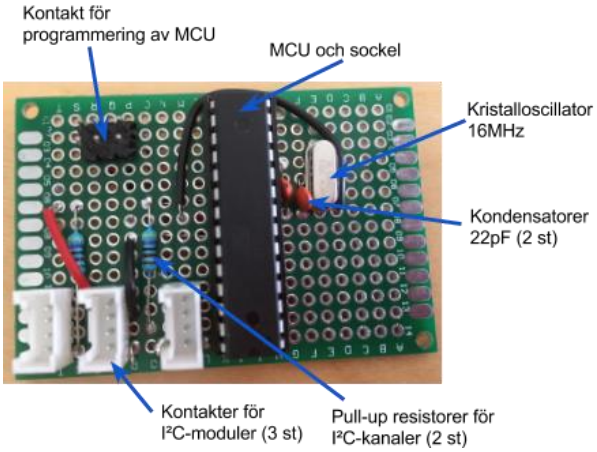
Figur 9 Lödda och urkopplade modulära komponenter

Ett en kasserad optisk mus användes som hölje för prototyp 2. Eltejp och smältlim användes för sammanfogning.

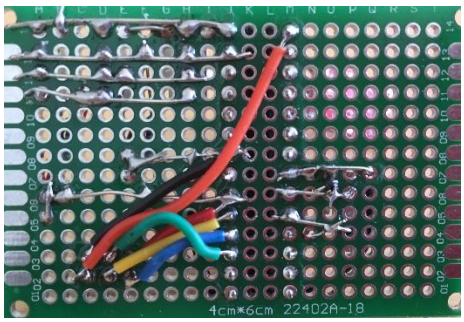
Figur 10 Prototyp 2 med hölje, framsida med display

Figur 11 Prototyp 2 med hölje, baksida med batteri

Efter sammanställning av prototyp 2, plockades delarna ut ur höljet, och en ytterligare anslutning löddes på kopplingsplattan, för programmering av mikroprocessorn via AVR pocket programmer.



Figur 12 Placering av fastlödda komponenter på kopplingsplatta



Figur 13 Komponenter lödda på kopplingsplatta, baksida

Prototyp 2 används som bas för den färdigställda produkten.

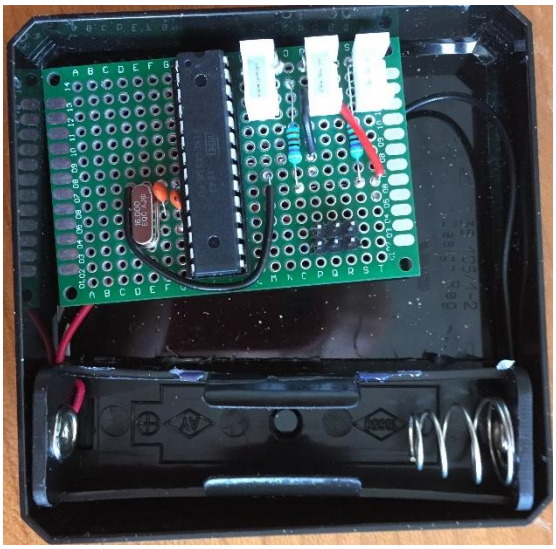
Kopplingsschema för prototyp 2 kan ses i rapportens bilagor.

2.6 Slutgiltig produkt

Elektronikmässigt är den färdiga produkten lik prototyp 2.

Skillnader är att koden har uppdaterats och utvecklats, och att dess hölje har bytts ut.

Höljet till den slutgiltiga produkten är byggt av en förvaringsbox för smycken. Ett uttag har gjorts i boxens lock, för inplacering av displayen. Displayen är fäst med tejp i locket, och övriga komponenter är fastsatta med smältlim i boxen.



Figur 14 Komponenter inplacerade i box

Figur 15 Komplett produkt

Produkten färdigställdes 2017-04-19

2.7 Stegräknarens funktion

Vid stegräknarens start kalibreras den genom att hämta 20 stycken värden för acceleration från MPU-9250-enheten. Stegräknaren bör ligga stilla under kalibreringen. Medelvärdet av de 20 värdena sätts till att vara stegräknarens värde för när den är passiv. Detta värde benämns som *idle*-värde.

Meddelandet "*Calibrating..*" visas på stegräknarens display under kalibreringen. När kalibreringen är färdig visas meddelandet "*Idle..*" på displayen, och stegräknaren kan börja användas.

Under användning hämtas accelerationsdata kontinuerligt från MPU-9250-enheten. Skillnaden mellan detta värde och *idle*-värdet beräknas som ett absolutvärde. Om skillnaden mellan dessa två värden överskrider ett visst satt tröskelvärde, så anses det att ett steg har tagits.

Vid varje taget steg uppdateras stegräknarens display med det totala antalet tagna steg.

Stegräknarens display sätts automatiskt i sömnläge om inget steg har tagits på ca 30 sekunder. Displayen vaknar automatiskt när ett steg tas.

2.8 Projektets kod

Utförligare kommentering av koden finns i källkodsfilen *main.c*.[3]

2.8.1 Egeninkluderade standardbibliotek

Bibliotek	Beskrivning
<avr/io.h>	Diverse I/O definitioner för mikrokontrollern.
<util/delay.h>	För funktionen *_delay_ms*() som pauserar programmet i angiver antal millisekunder.
<avr/interrupt.h>	Används för funktionen *sei(),* som aktiverar interrupt på mikrokontrollern. Behövs för kommunikation med MPU-9250.
<math.h>	Diverse matematikfunktioner och macron. Används för funktionen *fabs()*, som ger absolutvärdet av ett tal.
<stdio.h>	För funktionen *sprintf(),* som skriver en formaterad textsträng till en char-array. Används för att skriva ut stegräkning till display.

2.8.2 Inkluderade icke-standardbibliotek

Flera av dessa bibliotek har egna inkluderingar av olika standardbibliotek.

Bibliotek	Beskrivning
"i2chw/i2cmaster.h" *I2C master library using hardware TWI interface* Skapat av Peter Fleury	I^2C-funktionalitet för AVR-enheter. För kommunikation mellan mikrokontroller och enheter anslutna till dess I^2C-portar SDA/SCL.
"mpu6050/mpu6050.h" *MPU6050 lib 0x02* Skapat av Davide Gironi	För accelerometer-enheten, via I^2C. Viss egen kod har skrivits genom att bryta ut kod från detta bibliotek.
"u8g/u8g.h" *Universal 8bit Graphics Library* Skapat av olikraus	Funktionsbibliotek till AVR för diverse displayer.

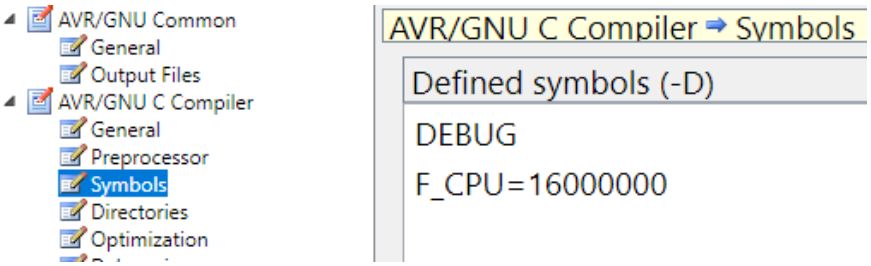
[3] För länk, se kapitel 1.4.1

2.8.3 Macron definierade i main.c

Macron som är definierade i main.c

Namn	Värde	Beskrivning
X	**0x3B**	Address för accelerometerdata i X-riktning i MPU-9250
Y	**0x3D**	Address för accelerometerdata i Y-riktning i MPU-9250
Z	**0x3F**	Address för accelerometerdata i Z-riktning i MPU-9250
STEP_ACC_TRIGGER	**2.3**	Tröskelvärde för acceleration, att bestämma om ett steg har tagits.
SET_IDLE_LOOP	**20**	Antal värden som ska hämtas från accelerometern för att bestämma "idle"-värde. Det värde som accelerometern ger när den anses ligga stilla.
DISPLAY_LINE_HEIGHT	**16**	Pixelhöjd för en rad på displayen. Används vid utskrift av text.
DISPLAY_MIDDLE	**(DISPLAY_LINE_HEIGHT * 2 + 2)**	Ungefärligt pixelvärde för utskrift på displayens mitt.
DISPLAY_SLEEP_DELAY	**3000**	När värdet i variabeln *displaySleepTimer* når detta värdet sätts displayen i sömnläge.

Macrot F_CPU behövs vara angivet för standardbiblioteket *delay.h,* detta sätts i kodprojektets egenskaper till *F_CPU=16000000*, i programmet Atmel Studio.



Figur 16 Kodprojektets egenskaper – Defined symbols

2.8.4 Macron från andra bibliotek

Macron som används i main.c, vilka är definierade i andra bibliotek.

Namn	Värde	Från bibliotek	Beskrivning
MPU6050_ADDR	**(0x68 <<1)**	*mpu6050.h*	I²C-address för MPU-9250
MPU6050_AGAIN	**16384.0**	*mpu6050.h*	Värde för att omvandla ”rå” accelerationsdata från MPU-9250 till g-krafter.
I2C_WRITE	**0**	*i2cmaster.h*	Datariktning för I²C - för att skriva till en enhet.
I2C_READ	**1**	*i2cmaster.h*	Datariktning för I²C - för att läsa från en enhet

2.8.5 Variabler

Variabler i källkoden, sätts i main.c

Namn	Beskrivning
uint16_t steps	Håller stegräkning.
double accIdle	Värde som sätts vid programmets uppstart, håller det värde som accelerometern ger när den ligger helt stilla.
double accCombined	Sammansatt accelerationsvärde från X- Y- och Z-riktning.
double accX, accY, accZ	Värden för acceleration i X- Y- och Z-riktning.
int displaySleeping	Håller displayens sömnläge, sätts till 1 om displayen är i sömnläge, annars 0.
int displaySleepTimer	Används som timer för displayens sömnläge. Ökar med 1 för varje varv i programmets huvudloop.
u8g_t u8g	Struct för biblioteket Universal 8bit Graphics Library, för display.

2.8.6 Funktioner

Egenskrivna funktioner i källkoden

Namn	Beskrivning
double getAcc(int addr);	Hämtar accelerationsdata från MPU och ger värdet i retur. Som argument tar funktionen adressen till det register som ska läsas. Dessa adresser är definierade som macron X, Y och Z.
void drawSteps(uint16_t steps);	Skriver ut aktuellt antal steg till displayens mitt, som *"Steps: x"*
void drawString(char * string, int line);	Skriver ut en text till displayen. Texten ges som argument till funktionen, i form av en pekare till en char-sträng. Line är den rad på displayen där texten ska visas.
void setAccIdle();	Sätter ett värde till den globala variabeln *accIdle*. Funktionen körs endast en gång vid varje uppstart av programmet.
double getAccXYZ(void);	Använder sig av funktionen *getAcc()* för att hämta accelerationsdata från X-, Y- och Z-riktningen. Det adderade värdet ges i retur.
void toggleDisplaySleep(void);	Sätter displayen i sömnläge, eller väcker displayen.
u8g_t u8g	Struct för biblioteket Universal 8bit Graphics Library, för display.

2.8.7 Main-funktionen

I detta kapitel beskrivs grundläggande vad som händer i programmets kod

Interrupt-funktionen aktiveras för MCU-enheten, detta behövs för att MPU-enheten ska fungera korrekt. Sedan initieras MPU-9250-enheten, med funktion från *mpu6050.h*

```
sei();
mpu6050_init();
```

Displayen initieras och teckensnitt bestäms, funktioner från *u8g.h*

```
u8g_InitI2C(&u8g, &u8g_dev_ssd1306_128x64_i2c, U8G_I2C_OPT_NONE);
u8g_SetFont(&u8g, u8g_font_fub14);
```

Texten ”STEPCOUNTER” visas på displayens mitt i en sekund.

```
drawString("STEPCOUNTER!", DISPLAY_MIDDLE);
_delay_ms(1000);
```

Idle-värde för accelerometern sätts.

```
setAccIdle();
```

Programmets huvud-loop startar.

```
while(1) {
```

Ett kombinerat accelerationsvärde för X- Y- och Z-riktningar hämtas

```
accCombined = getAccXYZ();
```

Om skillnaden mellan det hämtade värdet och idle-värdet överskrider tröskelvärdet för att trigga ett steg, skriv ut antalet steg till displayen. Om displayen är i sömnläge, väck den.

```
if(fabs(accCombined - accIdle) > STEP_ACC_TRIGGER){
    if(displaySleeping){
        toggleDisplaySleep();
    }
    drawSteps(steps++);
    _delay_ms(50);
}
```

Sätt display i sömnläge om timer har nått värde

```
if(displaySleepTimer++ > DISPLAY_SLEEP_DELAY){
    displaySleepTimer = 0;
    if(!displaySleeping){
        drawString("Sleeping", DISPLAY_MIDDLE);
        _delay_ms(1000);
        toggleDisplaySleep();
    }
}
```

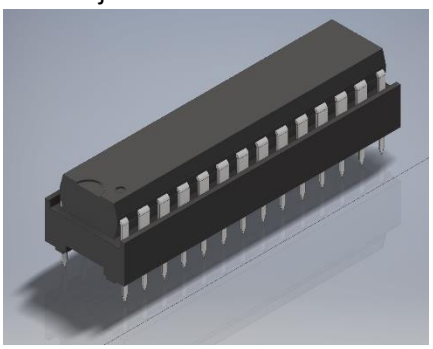
Slut på huvud-loop.

3 Diskussion och slutsats

I denna del utförs diskussioner och tankegångar angående projektets genomförande och resultat

CAD / 3D-utskrift av hölje

Initialt planerades att konstruera ett eget hölje med hjälp av en 3D-skrivare. Vilket hade gett en anpassad form för stegräknaren. På grund av tidsbrist valdes en redan befintlig produkt att användas som hölje.



Figur 17 Rendering av CAD-modell för MCU och sockel

Byte av batterityp

Batteriet av typ 18650 tar upp en mycket stor del av stegräknarens volym. Ett eventuellt byte till ett volymmässigt mindre batteri, exempelvis av typen CR2032, skulle reducera båda volym och vikt. Det är dock oklart om detta batteri kan användas för drivning av produkten.

Figur 18 Batteri av typen CR2032

Fler I²C-moduler

Stegräknaren har stöd för inkoppling av en tredje enhet. I projektets gång gjordes tester med WiFi-moduler av typen ESP8266, för att kunna kommunicera med stegräknaren trådlöst. Dock lyckades testerna och WiFi-stöd valdes att läggas ned.

En annan möjlig enhet skulle kunna vara en blåtand-modul.

Användning av färdigskrivna kodbibliotek

Flertalet redan färdiga bibliotek användes för att kommunicera med enheter som är kopplade till MCU-n. En framtida övning skulle kunna vara att försöka skriva egna funktioner för I²C-kommunikation.

Bättre detektion av steg
Stegräknaren anses i nuläget ej vara speciellt ackurat. Vid promenad detekteras oftast enbart steg på det ben där stegräknaren är placerad i byxfickan. Vid löpning, då större kraft utsöndras vid fotnedslag, detekteras båda benens steg.

Koden skulle kunna förbättras för att göra stegräknaren bättre.

Hastighet- och distans-mätning
Om stegräknaren hade tillgång till steglängden på den person som använder enheten, skull sträcka kunna mätas.

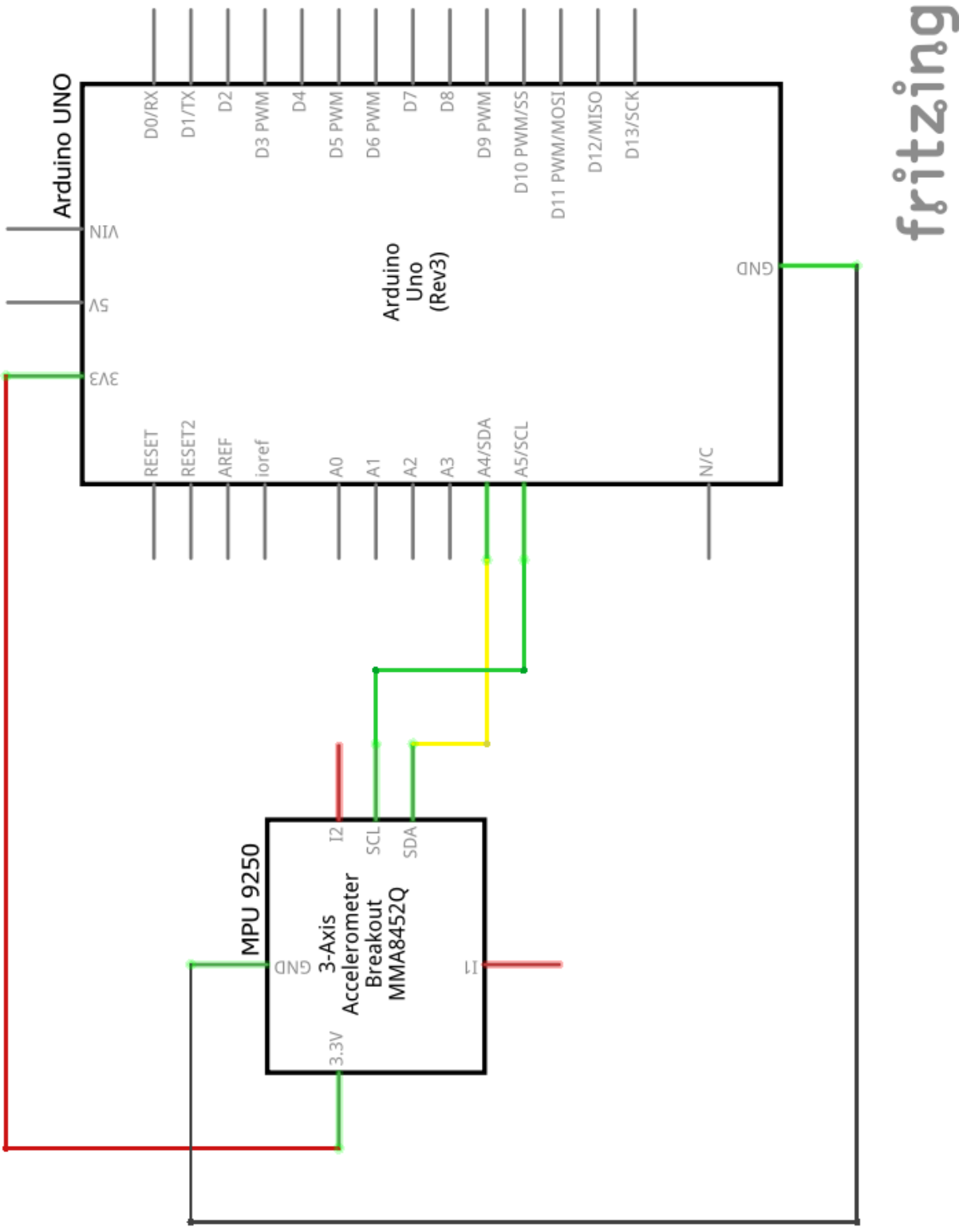
Även hastighet skulle kunna mätas, om stegräknaren hade tillgång till tid, t.ex. med en RTC-modul.

Avstängning
Stegräknaren är inte utrustad med någon strömbrytare, detta innebär att det enda sättet att stänga av eller nollställa enheten är att ta ur batteriet. Detta är ingen bra lösning.

På grund av tidsbrist och risk för att förstöra höljets lock vid uttag för strömbrytare valdes detta att inte implementeras.

4 Bilagor

4.1 Kopplingsschema för prototyp 1



4.2 Kopplingsschema för prototyp 2